**Java URLConnection and HttpURLConnection class:**

* These classes are used for developing java programs that communicate to server via URL(mostly using HTTP protocol).
* Able to write code for uploading and downloading files, webpages etc.
* And sending and retrieving responses etc.

**URLConnection class:**  is a super class of all that represent a connection between a java application and a URL. Provides API for generic URLs, and its subclass HttpURLConnection provides additional support for HTTP-specific features.

[Note: both are abstract classes-meaning that we can’t create new instances of URLConnection and HttpURLConnection, instead, an instance of URLConnection is obtained by opening the connection from URL object.

***openConnection()*** method of **URL class** establishes the connection.

public URLConnection openConnection() method -- returns the object of URLConnection class

getInputStream() method

e.g.

URL url=**new** URL(**"http://google.com"**);  
  
URLConnection urlcon=url.openConnection(); // instance is obtained on URL object.

or if http:// protocol, we can cast the returned object to HttpURLConnection object.

HttpURLConnection urlcon=(HttpURLConnection) url.openConnection();  
  
InputStream st=urlcon.getInputStream();

**HttpURLConnection class**: is http specific URLConnection, it work for HTTP protocols only.

Can be used to get information of any HTTP URL, such as header info, status code response body etc.

It is a subclass of URLConnection class.

URL url=**new** URL(**"https://www.eposnow.com/uk/careers"**);  
HttpURLConnection urlcon=(HttpURLConnection) url.openConnection();

for(int i=1;i<8;i++)

sout(urlcon.getHeaderFieldKey(i) + “ = ”+urlcon.getHeaderField(i));

}

urlcon.disconnect();

**Configure the URLConnection methods:**

* setConnectionTimeout(int timeout) sets connection timeout before connection can established.
* setReadTimeout(int timeout) sets read timeout in milliseconds
* setDoInput(Boolean doInput) -sets for reading content from server( default true).
* setDoInput(Boolean doInput) -sets for sending data to server( default false).

**Configure the HttpURLConnection methods:**

* setRequestMethod(String method) sets the method for URL request, which is one of HTTP methods GET, POST, HEAD, OPTIONS,PUT, DELETE and TRACE. ( default method is GET).
* setChunkedStreamingMode( int length) enables streaming of a HTTP request body.

The above methods are setters. And the URLConnection and HttpURLConnection classes also provide corresponding getters:

* getConnectTimeout()
* getReadTimeout()
* getDefaultUseCaches()
* getUseCaches()
* getDoInput()
* getDoOutput()
* getIfModifiedSince()
* getAllowUserInteraction()
* getDefaultAllowUserInteraction()
* getRequestProperty(String key)
* getRequestMethod()
* getFollowRedirects()
* getInstanceFollowRedirects()

**Read the Header fields:**

* once the connection is made the server processes the URL request and sends back a response that consists of metadata and actual content.
* The metadata is a collection of key=value pairs which are called header fields,
* Header fields reveal information about server, status code, protocol information, etc.
* The actual content can be in text, html, image etc.

**URLConnection** class provides methods for reading the header fields.

* getHeaderFields() returns a map that contains all header fields. (key=value pair).
* getHeaderField(int n) reads the vale of n-th header field.
* getHeaderField(String s) reads value of the named header field.
* getHeaderFieldKey(int n) reads they key of n-th header field.
* getHaederFieldDate(String s, long default) reads named field parsed date.

**The frequently accessed and specific methods:**

* getContentEncoding() reads value of the content-encoding header(encoding type of the content).
* getContentLenght() reads the value of the content-length header field( indicates size).
* getContentType() reads the value of content-type header field( type of content).
* getDate() reads the value of the date header field( date on server).

HttpURLConnection provides an additional method:

* getResponseCode() returns the HTTP status code sent by the server.

**Get an input stream and read data:** To read actual content, we need to obtain an *InputStream* instance from the connection abd use *InputStream* methods to read data.

e.g. [ to read data to array of bytes/ low level method to read.]

InputStream input=uc.getInputStream();

byte[] data=new byte[1024];

input.read(data);

e.g. [to read data to Characters, we have to wrap the InputStream in an InputStreaReader.]

InputStream input=uc.getInputStream();

InputStreamReader reader-new InputStreamReader(input);

int char=reader.read(); // reads a single character

char[] c=new Char[2096];

reader.read(c); // reads to an array of characters.

e.g. [ to read data to Strings, we have to wrap the InputStream in an BufferedReader.]

InputStream input=uc.getInputStream();

BufferedReader reader=new BufferedReader(new InputStreamReader(inout));

String str=reader.readLine(); // reads a line.

**Get an output stream and write data:**

To send data to server, we have to enable output on the connection first

urlcon.setOutput(true);

Then get OutputStream object associated to the connection. And use write() methods to write.

OutputStream output=urlcon.getOutputStream();

e.g [ to write data as an array of bytes.]

OutputStream output=urlcon.getOutputStream();

byte[] data=new byte[1024];

output,write(data);

e.g. [to write data as Character arrays, we have to wrap the Output Stream in an OutputStreamWriter.]

OutputStream output=urlcon.getOutputStream(new );

InputStreamReader writer=new OutputStreamWriter(output);

Int char=’a’;

writer.write(char); // write a single character

char[] c=new Char[2096];

writer.write(c); // writes an array of characters.

e.g. [ to write as Strings, we have to wrap the OutputStream in a PrinterWriter.]

PrintWriter writer=new PrintWriter(output);

String s=”hello world”;

writer.print(s);